DIGITAL DOWNLOAD हिंदी

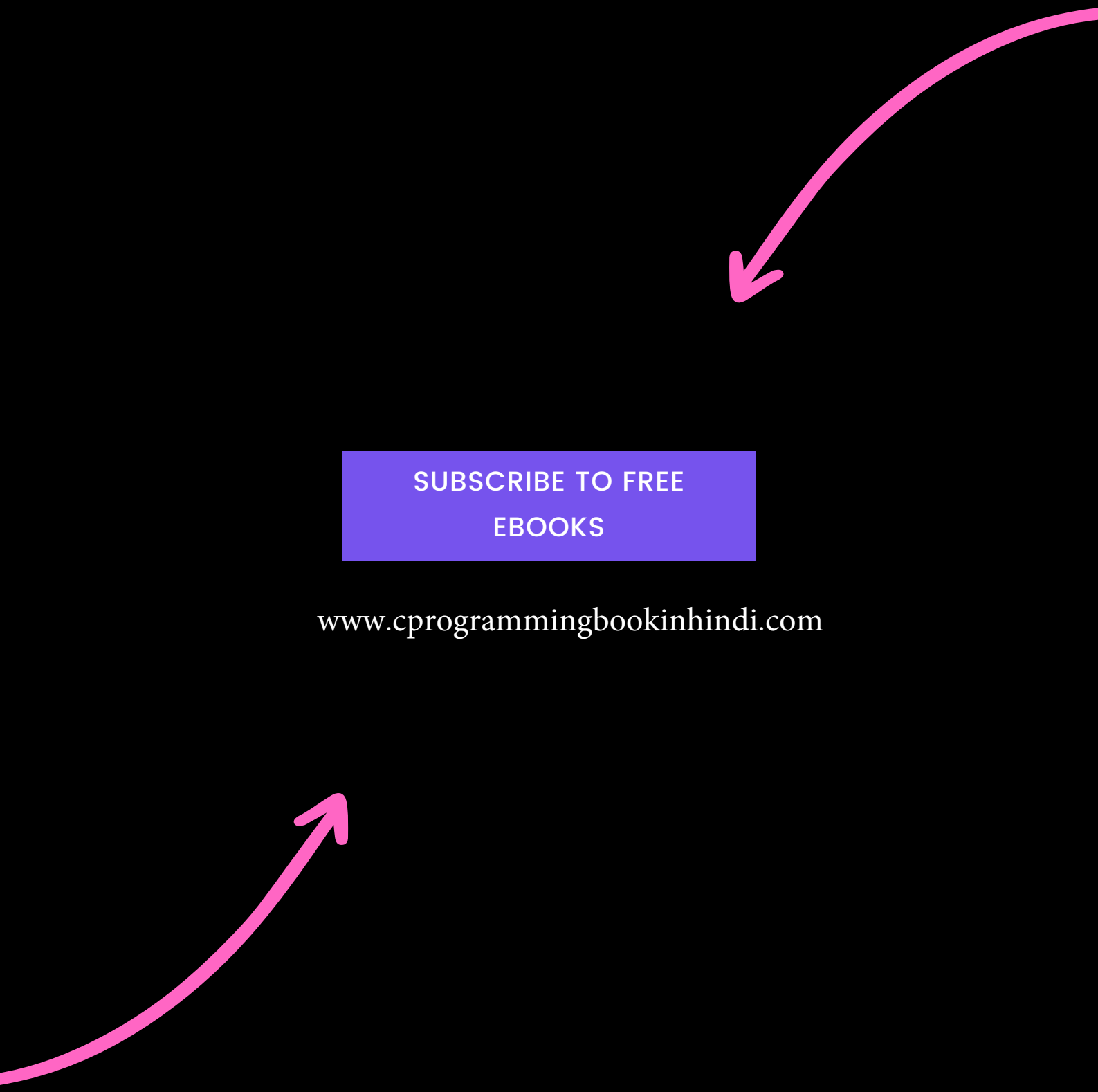
C# with OOP

Thanks for purchasing this Digital Download.

What's the success?

“We learn from failure, not from success!”
— Bram Stoker, Dracula

Please Keep this PDF in a safe place!


SUBSCRIBE TO FREE
EBOOKS
www.cprogrammingbookinhindi.com

Beginning C Sharp Object-Oriented Programming in Hindi

Copyright © 2017 by Machhindra Kalan

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Machhindra Kalan

Distributed to the book trade worldwide by Web Study Lab Pvt Ltd, Location - Mumbai, Palava city Pin 421204

E-mail: machhi29@gmail.com, or

Visit: www.cprogrammingbookinhindi.com

For information on translations, please contact Web Study Lab Pvt Ltd, Location - Mumbai, Palava city Pin 421204

Phone +91 86555-67778

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, the author shall not have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this book.

This book
is dedicated
to those

who really wants
to be a

Beginning C Sharp Object-Oriented Programming in Hindi

INDEX OF CONTENTS

Table of Contents

Introduction

Chapter 1: Overview of Object-Oriented Programming.....17

The History of OOP

Why Use OOP?

The Characteristics of OOP

Objects

Abstraction

Encapsulation

Polymorphism

Inheritance

Aggregation

The History of C#

Summary

Chapter 2: Designing OOP Solutions: Identifying the Class Structure….31

Goals of Software Design

Understanding the Unified Modeling Language

Developing a SRS

Introducing Use Cases

Understanding Class Diagrams

Modeling Object Relationships

Association

Inheritance

Aggregation

Association Classes

Summary

Chapter 3: Designing OOP Solutions: Modeling the Object Interaction...49

Understanding Scenarios

Introducing Sequence Diagrams

Message Types

Recursive Messages

Message Iteration
Message Constraints
Message Branching
Understanding Activity Diagrams
Decision Points and Guard Conditions
Parallel Processing.
Activity Ownership
Exploring GUI Design
GUI Activity Diagrams
Interface Prototyping
Interface Flow Diagrams
Application Prototyping
Summary
Chapter 4: Designing OOP Solutions: A Case Study……..69
Developing an OOP Solution
Creating the System Requirement Specification
Developing the Use Cases
Diagramming the Use Cases
Developing the Class Model
Identifying the Classes
Adding Attributes to the Classes
Identifying Class Associations
Modeling the Class Behaviors
Developing the User Interface Model Design
Avoiding Some Common OOP Design Pitfalls
Summary
Chapter 5: Introducing the .NET Framework and Visual Studio…….97
Introducing the .NET Framework
Goals of the .NET Framework
Support of Industry Standards
Extensibility
Unified Programming Models
Easier Deployment

Improved Memory Management
Improved Security Model
Components of the .NET Framework
Common Language Runtime
Framework Base Class Library
Data Classes80
Windows Applications
Web Applications
Application Services
Working with the .NET Framework
Understanding Assemblies and Manifests
Referencing Assemblies and Namespaces
Compiling and Executing Managed Code
Using the Visual Studio Integrated Development Environment
Customizing the IDE
Creating a New Project
Investigating the Solution Explorer and Class View
Exploring the Toolbox and Properties Window
Building and Executing the Assembly
Stepping Through Code
Setting Conditional Breakpoints
Locating and Fixing Build Errors.
Summary
Chapter 6: Creating Classes……..115
Introducing Objects and Classes
Defining Classes
Creating Class Properties
Creating Class Methods
Defining the Employee Class
Testing the Employee Class
Using Constructors
Overloading Methods
Creating and Overloading Class Constructors

Testing the Employee Class Constructors
Overloading a Class Method
Testing the Overloaded Update Method
Summary
Chapter 7: Creating Class Hierarchies……128
Understanding Inheritance
Creating Base and Derived Classes
Creating a Sealed Class
Creating an Abstract Class
Using Access Modifiers in Base Classes
Overriding the Methods of a Base Class
Calling a Derived Class Method from a Base Class
Calling a Base Class Method from a Derived Class
Overloading Methods of a Base Class
Hiding Base Class Methods
Implementing Interfaces
Understanding Polymorphism
Summary
Chapter 8: Implementing Object Collaboration……146
Communicating Through Messaging
Defining Method Signatures
Passing Parameters
Understanding Event-Driven Programming
Understanding Delegation
Implementing Events
Responding To Events
Windows Control Event Handling
Handling Exceptions in the .NET Framework
Using the Try-Catch Block
Adding a Finally Block
Throwing Exceptions
Nesting Exception Handling
Static Properties and Methods

Using Asynchronous Messaging
Summary

Chapter 9: Working with Collections……170
Introducing the .NET Framework Collection Types
Working with Arrays and Array Lists
Using Generic Collections
Programming with Stacks and Queues
Summary
Chapter 10: Implementing the Data Access Layer……186
Introducing ADO.NET
Working with Data Providers
Establishing a Connection
Executing a Command
Using Stored Procedures
Using the DataReader Object to Retrieve Data
Using the DataAdapter to Retrieve Data
Working with DataTables and DataSets
Populating a DataTable from a SQL Server Database
Populating a DataSet from a SQL Server Database
Establishing Relationships between Tables in a DataSet
Editing Data in the DataSet
Working with the Entity Framework
Querying Entities with LINQ to EF
Updating Entities with the Entity Framework
Summary
Chapter 11: Developing Windows Applications……219
Windows Fundamentals
Introducing XAML
Using Layout Controls
Adding Display Controls
Using the Visual Studio Designer
Handling Control Events

Creating and Using Dialog Boxes
Presenting a MessageBox to the User
Creating a Custom Dialog Box
Data Binding in Windows-Based GUIs
Binding Controls Using a DataContext
Creating and Using Control and Data Templates
Summary
Chapter 12: Developing Web Applications…….244
What Is Silverlight?
Creating a Silverlight Application
Using Layout Controls
Adding Display Controls
Handling Control Events
Data Binding in Silverlight
Validating and Converting Data
Summary
Chapter 13: Developing and Consuming WCF Services…….260
What Are Services?
Creating a WCF Web Service
Consuming a WCF Web Service.
Using Data Contracts
WCF Data Services
Summary
Chapter 14: Developing the OSO Application…….275
Revisiting Application Design
Building the OSO Application's Data Access and Business Logic Layers
Creating the OSO Application UI
Summary
Chapter 15: Wrapping Up……..315
Improve Your Object-Oriented Design Skills
Investigate the .NET Framework Namespaces
Become Familiar with ADO.NET and the Entity Framework

Learn More About WPF and Silverlight
Move Toward Component-Based Development
Find Help
Join a User Group
Please Provide Feedback
Thank You and Good Luck
Appendix A: Fundamental Programming Concepts
Working with Variables and Data Types
Understanding Elementary Data Types
Integral Data Types
Non-Integral Data Types
Character Data Types
Boolean Data Type
Date Data Type
Object Data Type
Nullable Types
Introducing Composite Data Types
Structures
Arrays
Classes
Looking at Literals, Constants, and Enumerations
Literals
Constants
Enumerations
Exploring Variable Scope
Block-Level Scope
Procedure Scope
Module Scope
Understanding Data Type Conversion
Implicit Conversion.
Explicit Conversion
Widening and Narrowing Conversions
Working with Operators

Arithmetic Operators
Comparison Operators
Logical Operators
Ternary Operator
Introducing Decision Structures
If Statements
Switch Statements
Using Loop Structures
While Statement
Do-While Statement
For Statement
For Each Statement
Introducing Methods
Appendix B: Exception Handling in C#
Managing Exceptions
Using the .NET Framework Exception Classes
Appendix C: Installing the Required Software
Installing the Sample Databases
Verifying the Database Installs
Congratulations!!!

INTRODUCTION

परिचय

यह एक के रूप में मेरा अनुभव रहा है .Net ट्रेनर और लीड प्रोग्रामर कि ज्यादातर लोगों को भाषा के syntax को चुनने में परेशानी नहीं होती है। कई लोगों को क्या चिंता और निराशा होती है, object-oriented प्रोग्रामिंग पद्धति और डिजाइन की उच्च–स्तरीय अवधारणाएं हैं। समस्या को कंपाउंड करने के लिए, अधिकांश परिचयात्मक प्रोग्रामिंग बुक्स और ट्रेनिंग क्लासेस इन अवधारणाओं को छोड़ देते हैं या बदतर, कवर उन्हें बिल्कुल नहीं करते हैं। मेरी यह आशा है कि यह पुस्तक इस शून्य को भर दे। इस पुस्तक को लिखने में मेरा लक्ष्य दुगना है। किसी कार्य के लिए पहला होना आपको C# में प्रोग्रामिंग की बुनियादी बातों को समझने के लिए आवश्यक जानकारी प्रदान करता है। दूसरी और अधिक महत्वपूर्ण बात, आपको object-oriented प्रोग्रामिंग पद्धति और डिजाइन की उच्च–स्तरीय अवधारणाओं को मास्टर करने के लिए आवश्यक जानकारी के साथ प्रस्तुत करना।

यह पुस्तक आपको एक व्यावसायिक समस्या को हल करने के उद्देश्य से एक object-oriented प्रोग्रामिंग समाधान को आर्किटेक्ट करने के लिए आवश्यक ज्ञान प्रदान करती है। जैसा कि आप पुस्तक के माध्यम से अपना काम करते हैं, आप पहले सीखेंगे कि किसी एपलिकेशन की व्यावसायिक आवश्यकताओं का विश्लेषण कैसे करें। अगला, आप समाधान डिजाइन में शामिल वस्तुओं और संबंधों को मॉडल करेंगे।

अंत में, आप C # का उपयोग करके समाधान को लागू करेंगे। रास्ते के साथ, आप सॉफ्टवेयर डिजाइन, यूनिफाइड मॉडलिंग लैंग्वेज (UML), आपत्तिजनक प्रोग्रामिंग, C # और .NET फ्रेमवर्क के मूल सिद्धांतों के बारे में जानेंगे। क्योंकि यह एक परिचयात्मक पुस्तक है, यह आपके द्वारा प्रस्तुत किए जाने वाले विषयों के आपके अध्ययन के लिए एक प्रारंभिक बिंदु है। इस प्रकार, यह पुस्तक आपको ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग और एक विशेषज्ञ बनाने के लिए डिज़ाइन नहीं की गई है।

न ही यह एक गहन अध्ययन दृश्य स्टूडियो है। इनमें से किसी एक क्षेत्र

में प्रवीण बनने में काफी समय और मेहनत लगती है। यह मेरी आशा है कि इस पुस्तक को पढ़ने से, object-oriented प्रोग्रामिंग में आपका पहला अनुभव होगा सुखद और बोधगम्य - और ये अनुभव आगे के अध्ययन की इच्छा पैदा करेंगे।

Target Audience

इस पुस्तक के लिए लक्षित श्रोता शुरुआत C# प्रोग्रामर है जो एक नींव हासिल करना चाहता है। C# भाषा की मूल बातें के साथ ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग। एक से संक्रमण प्रोग्रामर प्रक्रिया–procedural प्रोग्रामिंग मॉडल से ऑब्जेक्ट–ओरिएंटेड मॉडल भी इस पुस्तक से लाभानवित होंगे। इसके अलावा, कई विज़ुअल बेसिक (VB) प्रोग्रामर हैं। जो C # को संक्रमण करना चाहते हैं। इससे पहले C# के लिए संक्रमण, यह अनिवार्य है कि आप ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग की बुनियादी बातों को समझें। क्योंकि एक "शुरुआत" का अनुभव स्तर काफी भिन्न हो सकता है, मैंने एक प्राइमर को शामिल किया है परिशिष्ट A जिसमें कुछ बुनियादी प्रोग्रामिंग अवधारणाओं पर चर्चा की गई है और उन्हें C# में कैसे लागू किया गया है। मैं यदि आप प्रोग्रामिंग के लिए नए हैं, तो आप इन अवधारणाओं की समीक्षा करने का सुझाव देंगे।

Organization of the Book

इस पुस्तक को तीन भागों में व्यवस्थित किया गया है:

PART 1

ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग कार्यप्रणाली और डिजाइन–अवधारणा में पार कर जाता है जो पार कर जाता है एक विशेष प्रोग्रामिंग भाषा। प्रस्तुत अवधारणाओं को क्रियानवित भाषा की परवाह किए बिना एक आपततिजनक प्रोग्रामिंग समाधान की सफलता के लिए महत्वपूर्ण है। इस भाग के समापन पर, एक केस स्टडी आपको वास्तविक दुनिया में मॉडलिंग के कदमों के माध्यम से चलता है।

PART 2

यह देखता है कि C# में ऑब्जेक्ट ओरिएंटेड प्रोग्रामिंग कैसे लागू की जाती है। आप बनाते हुए देखेंगे श्रेणी संरचनाएं, पदानुक्रम बनाना और इंटरफ़ेस लागू करना। यह भाग वस्तु का भी बातचीत और सहयोग परिचय देता है। आप देखेंगे कि ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग विषयों में कैसे चर्चा की गई है। भाग 1 C# कोडिंग निर्माण में बदल जाता है।

PART 3

भाग 3 कवर बनाना .NET application। आप सीखेंगे कि डेटा एक्सेस लेयर को कैसे विकसित किया जाए कक्षाएं जो ADO.NET बनाती हैं। नेमस्पेस का नेट सेट। आप एक विंडोज–आधारित यूजर इंटरफेस बनाएंगे वेब–आधारित उपयोगकर्ता इंटरफ़ेस, और सेवा–आधारित प्रोग्रामेटिक इंटरफ़ेस। भाग 3 के अंत में, आप करेंगे भाग 1 में विकसित केस स्टडी को फिर से देखें और डिज़ाइन को पूरी तरह कार्यात्मक C# एप्लिकेशन में बदलें। इसमें एक ग्राफिकल यूजर इंटरफेस बनाना, बिजनेस लॉजिक को लागू करना और ए के साथ इंटीग्रेट करना शामिल है डेटा संग्रहीत करने के लिए relational database डेटाबेस।

Activities and Software Requirements

सीखने के सबसे महत्वपूर्ण पहलुओं में से एक है। आप बिना कूदें बाइक चलाना नहीं सीख सकते और आप कोड को क्रैंक किए बिना प्रोग्राम करना नहीं सीख सकते। कोई भी सफल प्रशिक्षण program एक सिद्धांत घटक और एक हाथों पर घटक दोनों को शामिल करने की आवश्यकता है।

मैंने इस पूरी पुस्तक में दोनों घटकों को शामिल किया है। यह मेरी आशा है कि आप इसे गंभीरता से लेंगे जिन गतिविधियों को मैंने प्रत्येक अध्याय में जोड़ा है और उनके माध्यम से पूरी तरह से काम करता हूं – यहां तक कि बार–बार। कुछ छात्रों की धारणा के विपरीत कि ये गतिविधियाँ "टाइपिंग में अभ्यास" हैं।

सिद्धांत को ठोस बनाने का मौका और जहाँ अवधारणाओं का सही अनुकरण होता है। मैं भी गतिविधि के माध्यम से काम करने के लिए प्रोत्साहित करें। कोड में से कुछ को बदलने से डरो मत देखते हैं क्या होता है। सीखने के कुछ बेहतरीन अनुभव तब होते हैं जब छात्र लाइनों के बाहर रंग जमाते हैं।

भाग 1 में UML मॉडलिंग गतिविधियाँ UMLet का उपयोग करने वाले किसी व्यक्ति के लिए हैं। मैंने यह program इसलिए चुना क्योंकि यह सीखने के लिए एक अच्छा चित्र है। यह आपको बहुत कुछ जोड़े बिना UML चित्र बनाने देता है उन्नत सुविधाएँ उच्च अंत CASE उपकरणों से जुड़ी हैं। UMLet एक फ्री ओपन सोर्स टूल है और हो सकता है www.umlet.com से डाउनलोड किया गया। गतिविधियों को पूरा करने के लिए आप Visio जैसे किसी अन्य टूल का भी उपयोग कर सकते हैं। हालाँकि, आपको इन गतिविधियों को पूरा करने के लिए एक उपकरण की भी आवश्यकता नहीं है; कागज और पेंसिल ठीक काम करेगा।

भाग 2 में गतिविधियों को C# स्थापित के साथ विजुअल स्टूडियो 2011 एक्सप्रेस की आवश्यकता है। मैं आपको प्रोत्साहित करता हूं गतिविधियों को पूरा करने के दौरान मदद फ़ाइलों को स्थापित करें और उनका पर्याप्त उपयोग करें। भाग 3 में गतिविधियाँ Pubs और नॉर्थविंड डेटाबेस स्थापित के साथ Microsoft SQL Server 2008 की आवश्यकता है। परिशिष्ट सी नमूना डेटाबेस को डाउनलोड करने और स्थापित करने के निर्देश शामिल हैं। आप मुफ्त एक्सप्रेस पा सकते हैं दोनों विजुअल स्टूडियो 2010 और SQL erver 2008 के संस्करण www.msdn.microsoft.com पर।

CHAPTER 1

OVERVIEW OF OBJECT ORIENTED PROGRAMMING

ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग और C# के अपने अध्ययन के लिए चरण निर्धारित करने के लिए, यह अध्याय संक्षेप में दिखेगा ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग और ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग की विशेषताओं का इतिहास भाषा: हिन्दी। आप देखेंगे कि ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग औद्योगिक–शक्ति वितरित सॉफ्टवेयर सिस्टम के विकास में इतनी महत्वपूर्ण क्यों हो गई है। आप यह भी जांच करेंगे कि C# कैसे अग्रणी एपलिकेशन प्रोग्रामिंग भाषाओं में से एक में विकसित है ।

इस अध्याय को पढ़ने के बाद, आप निम्नलिखित से परिचित होंगे:

- ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग क्या है।
- क्यों ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग विकास में इतनी औद्योगिक शक्ति applications के महत्वपूर्ण हो गई है।
- एक भाषा को object-oriented बनाने वाली विशेषताएँ।
- C# का इतिहास और विकास।

The History of OOP

ऑब्जेक्ट–ओरिएंटेड प्रोग्रामिंग (OOP) सॉफ्टवेयर डेवलपमेंट के लिए एक दृष्टिकोण है जिसमें संरचना सॉफ्टवेयर एक कार्य को पूरा करने के लिए एक दूसरे के साथ बातचीत करने वाली वस्तुओं पर आधारित है। यह परस्पर क्रिया होती है वस्तुओं के बीच आगे और पीछे से गुजरने वाले संदेशों का रूप। एक संदेश के जवाब में, एक वस्तु एक क्रिया या method कर सकते हैं।

यदि आप देखते हैं कि आप अपने आस–पास की दुनिया में कार्यों को कैसे पूरा करते हैं, तो आप देख सकते हैं कि आप एक में बातचीत करते हैं object-oriented दुनिया। यदि आप स्टोर पर जाना चाहते हैं, उदाहरण के लिए, आप कार ऑब्जेक्ट के साथ इंटरैक्ट करते हैं। एक गाडी ऑब्जेक्ट में अन्य ऑब्जेक्ट्स होते हैं जो आपको प्राप्त करने के कार्य को पूरा करने

के लिए एक दूसरे के साथ बातचीत करते हैं। आप कुंजी को इग्निशन ऑब्जेक्ट में डालें और इसे चालू करें। यह बदले में एक संदेश भेजता है (एक के माध्यम से) स्टार्टर ऑब्जेक्ट को इलेक्ट्रिकल सिग्नल), जो कार को शुरू करने के लिए इंजन ऑब्जेक्ट के साथ इंटरैक्ट करता है। ड्राइवर के रूप में, आप इस तर्क से अलग हैं कि सिस्टम की वस्तुएं कार को शुरू करने के लिए एक साथ कैसे काम करती हैं। तुम बस कुंजी के साथ इग्निशन ऑब्जेक्ट की शुरुआत method को निष्पादित करके घटनाओं के अनुक्रम को आरंभ करें। आप फिर सफलता या विफलता की प्रतिक्रिया (संदेश) की प्रतीक्षा करें। इसी तरह, सॉफ्टवेयर प्रोग्राम के उपयोगकर्ता किसी कार्य को पूरा करने के लिए आवश्यक तर्क से अलग–थलग के लिये हैं। उदाहरण, जब आप अपने वर्ड प्रोसेसर में एक पेज प्रिंट करते हैं, तो आप एक प्रिंट बटन पर क्लिक करके कार्रवाई शुरू करते हैं। आप आंतरिक प्रसंस्करण से अलग हो जाते हैं जो होने की आवश्यकता होती है; आप बस एक प्रतिक्रिया की प्रतीक्षा करें आपको बता रहा है कि क्या यह मुद्रित है। आंतरिक रूप से, बटन ऑब्जेक्ट एक प्रिंटर ऑब्जेक्ट के साथ इंटरैक्ट करता है, जिसके साथ इंटरैक्ट करता है पृष्ठ को प्रिंट करने के कार्य को पूरा करने के लिए प्रिंटर।

ओओपी अवधारणाओं ने 1960 के दशक के मध्य में सिमूला और एक प्रोग्रामिंग भाषा के साथ सरफेसिंग शुरू की आगे 70 के दशक में स्मालटाक के आगमन के साथ विकसित हुआ। हालांकि सॉफ्टवेयर डेवलपर्स ने नहीं किया ओओपी भाषाओं, ऑब्जेक्ट–ओरिएंटेड कार्यप्रणाली में इन शुरुआती प्रगति को भारी रूप से स्वीकार करते हैं विकसित करना जारी रखा। 80 के दशक के मध्य में object-oriented पद्धति में रुचि का पुनरुत्थान हुआ।

विशेष रूप से, C और Eifle प्रोग्रामर जैसी OOP भाषाएँ मुख्यधारा के कंप्यूटर से लोकप्रिय हुईं। ओओपी 90 के दशक में लोकप्रियता में वृद्धि करता रहा, विशेष रूप से जावा और के आगमन के साथ विशाल इसके बाद आकर्षित हुआ। और 2002 में, के रिलीज के साथ संयोजन के रूप

में। Microsoft ने एक नई OOP भाषा, C# (सी–शार्प उच्चारण की) और विजुअल बेसिक को फिर से शुरू किया यह वास्तव में एक ओओपी भाषा है।

Why Use OOP?

OOP आज व्यावसायिक समस्याओं के समाधान के लिए इतने व्यापक रूप से इस्तेमाल किए जाने वाले प्रतिमान में क्यों विकसित हुआ है? दौरान 70 और 80 के दशक में, सी, पास्कल और फोरट्रान जैसी procedural-oriented प्रोग्रामिंग भाषाएं व्यापक रूप से थीं व्यापार–उन्मुख सॉफ्टवेयर सिस्टम विकसित करने के लिए उपयोग किया जाता है। प्रक्रियात्मक भाषाएं program को एक में व्यवस्थित करती हैं रैखिक फैशन – वे ऊपर से नीचे तक चलते हैं। दूसरे शब्दों में, program एक चलने वाले चरणों की एक श्रृंखला है दूसरे के बाद। इस तरह की प्रोग्रामिंग ने छोटे programs के लिए ठीक काम किया, जिसमें कुछ सौ कोड लाइन्स शामिल थे, लेकिन जैसे–जैसे program बड़े होते गए, उन्हें प्रबंधित करना और डीबग करना कठिन होता गया। programs के लगातार बढ़ते आकार को प्रबंधित करने के प्रयास में, संरचित प्रोग्रामिंग थी कार्यों या प्रक्रियाओं नामक प्रबंधनीय खंडों में कोड को तोड़ने के लिए शुरू किया गया। यह था एक सुधार, लेकिन जैसा कि programs ने अधिक जटिल व्यावसायिक कार्यक्षमता का प्रदर्शन किया और उनके साथ बातचीत की अन्य प्रणालियाँ, संरचनात्मक प्रोग्रामिंग पद्धति की निम्न खामियाँ सतह पर आने लगीं:

- Program बनाए रखना कठिन हो गया।
- मौजूदा कार्यक्षमता में सभी को प्रतिकूल रूप से प्रभावित किए बिना सिस्टम की कार्यक्षमता परिवर्तन करना कठिन था।
- नए programs को अनिवार्य रूप से खरोंच से बनाया गया था। नतीजतन, पिछले प्रयासों के निवेश पर वापसी बहुत कम था।

- प्रोग्रामिंग टीम के विकास के लिए अनुकूल नहीं था। प्रोग्रामर को करना पड़ा एक program कैसे काम करता है और उनके प्रयासों को अलग नहीं कर सकता है के हर पहलू को जानें एक प्रणाली का एक पहलू।

- यवसाय मॉडल को प्रोग्रामिंग मॉडल में अनुवाद करना कठिन था।

- इसने अलगाव में अच्छा काम किया लेकिन अन्य प्रणालियों के साथ अच्छी तरह से एकीकृत नहीं किया। इन कमियों के अलावा, कंप्यूटिंग सिस्टम के कुछ विकास के कारण आगे तनाव पैदा हो गया संरचनात्मक program दृष्टिकोण, जैसे कि:

- नॉनप्रोग्रामर्स ने मांग की और programs के माध्यम से सीधे पहुंच दी गई ग्राफिकल यूजर इंटरफेस और उनके डेस्कटॉप कंप्यूटर का समावेश।

- उपयोगकर्ताओं ने बातचीत करने के लिए अधिक सहज, कम संरचित दृष्टिकोण program की मांग की।

- कंप्यूटर सिस्टम एक वितरित मॉडल में विकसित हुआ, जहां व्यापार तर्क, उपयोगकर्ता इंटरफ़ेस, और बैकएंड डेटाबेस को बुरी तरह से युग्मित किया गया और इंटरनेट और इंट्रानेट पर एक्सेस किया गया।

परिणामस्वरूप, कई व्यावसायिक सॉफ़्टवेयर डेवलपर ऑब्जेक्ट–ओरिएंटेड कार्यप्रणाली में बदल गए और प्रोग्रामिंग भाषा इन समस्याओं को हल करने के लिए। लाभों में निम्नलिखित शामिल थे:

- व्यापार विश्लेषण मॉडल से सॉफ्टवेयर तक एक अधिक सहज संक्रमण

  कार्यान्वयन मॉडल।

- programs में परिवर्तनों को अधिक कुशलता से बनाए रखने और

लागू करने की क्षमता और तेजी से।

- टीम प्रक्रिया का उपयोग करके सॉफ़्टवेयर सिस्टम को अधिक प्रभावी ढंग से बनाने की क्षमता, विशेषज्ञों को सिस्टम के कुछ हिस्सों पर काम करने की अनुमति देना।
- अन्य programs और खरीद में कोड घटकों का पुन: उपयोग करने की क्षमता की कार्यक्षमता बढ़ाने के लिए तृतीय–पक्ष डेवलपर्स द्वारा लिखे गए घटक थोड़े प्रयास से उनके program।
- शिथिल युग्मित कंप्यूटिंग सिस्टम के साथ बेहतर एकीकरण।
- आधुनिक ऑपरेटिंग सिस्टम के साथ बेहतर एकीकरण।
- उपयोगकर्ताओं के लिए अधिक सहज चित्रमय उपयोगकर्ता इंटरफ़ेस बनाने की क्षमता।

The Characteristics of OOP

इस खंड में आप सभी ओओपी भाषाओं के लिए कुछ मूलभूत अवधारणाओं और शर्तों को देखने जा रहे हैं। किसी भी विशेष प्रोग्रामिंग भाषा में इन अवधारणाओं को कैसे लागू किया जाए, इसके बारे में चिंता न करें; वह बाद में आएगा।

मेरा लक्ष्य आपको केवल अवधारणाओं से परिचित करना है और उन्हें अपने रोजमर्रा के अनुभवों से इस तरह से जोड़ना है कि वे बाद में और अधिक समझ में आते हैं जब आप ओओपी डिजाइन और कार्यान्वयन को देखते हैं।

Objects

जैसा कि मैंने पहले उल्लेख किया है, हम एक object-oriented दुनिया में रहते हैं। आप एक वस्तु हैं। आप अन्य वस्तुओं के साथ बातचीत करते हैं। वास्तव में, आप डेटा के साथ एक वस्तु हैं जैसे कि ऊँचाई और बालों

का रंग। आपके पास ऐसे तरीके भी हैं जो आप पर प्रदर्शन करते हैं या आप पर किए जाते हैं, जैसे कि खाना और चलना।

तो वस्तुएं क्या हैं? OOP शब्दों में, एक ऑब्जेक्ट डेटा को शामिल करने और उस डेटा के साथ काम करने की प्रक्रियाओं के लिए एक संरचना है। उदाहरण के लिए, यदि आप इन्वेंट्री में उत्पादों से जुड़े डेटा को ट्रैक करने में रुचि रखते हैं, तो आप एक उत्पाद वस्तु बनाएंगे जो उत्पादों से संबंधित डेटा को बनाए रखने और काम करने के लिए जिम्मेदार है।

यदि आप अपने आवेदन में मुद्रण क्षमता चाहते हैं, तो आप एक प्रिंटर ऑब्जेक्ट के साथ काम करेंगे जो आपके प्रिंटर के साथ बातचीत करने के लिए उपयोग किए जाने वाले डेटा और methodयों के लिए जिम्मेदार है।

Abstraction

जब आप दुनिया में वस्तुओं के साथ बातचीत करते हैं, तो आप अक्सर उनके गुणों के सबसेट से चिंतित होते हैं। वस्तुओं के बाहरी गुणों को abstract या फ़िल्टर करने की इस क्षमता के बिना, आपको सूचना बमबारी की अधिकता को संसाधित करने और हाथ में काम पर ध्यान केंद्रित करने में मुश्किल होगी। abstraction के परिणामस्वरूप, जब दो अलग–अलग लोग एक ही वस्तु के साथ बातचीत करते हैं, तो वे अक्सर विशेषताओं के एक अलग सबसेट के साथ व्यवहार करते हैं।

जब मैं अपनी कार चलाता हूं, उदाहरण के लिए, मुझे कार की गति और उस दिशा की दिशा को जानना होगा। क्योंकि कार एक स्वचालित है, इसलिए मुझे इंजन के आरपीएम को जानने की आवश्यकता नहीं है, इसलिए मैं इस जानकारी को फ़िल्टर करता हूं। दूसरी ओर, यह जानकारी एक रेसकार चालक के लिए महत्वपूर्ण होगी, जो इसे फ़िल्टर नहीं करेगा।

ओओपी applicatins में वस्तुओं का निर्माण करते समय, abstraction की

इस अवधारणा को शामिल करना महत्वपूर्ण है। यदि आप एक शिपिंग एपलिकेशन का निर्माण कर रहे थे, तो आप आकार और वजन जैसी विशेषताओं के साथ एक उत्पाद ऑब्जेक्ट का निर्माण करेंगे।

आइटम का रंग बाहरी जानकारी और फ़िल्टर किया हुआ होगा। दूसरी ओर, ऑर्डर–एंट्री एपलिकेशन का निर्माण करते समय, रंग महत्वपूर्ण और हो सकता है उत्पाद वस्तु की विशेषता के रूप में शामिल किया जाएगा।

Encapsulation

ओओपी की एक अन्य महत्वपूर्ण विशेषता एनकैप्सुलेशन है। एनकैप्सुलेशन वह प्रक्रिया है जिसमें कोई प्रत्यक्ष नहीं होता है। डेटा तक पहुंच दी जाती है; इसके बजाय, यह छिपा हुआ है। यदि आप डेटा तक पहुंच प्राप्त करना चाहते हैं, तो आपको करना होगा डेटा के लिए जिम्मेदार वस्तु के साथ बातचीत। पिछले इन्वेंट्री उदाहरण में, यदि आप चाहते थे उत्पादों पर जानकारी देखने या अद्यतन करने के लिए, आपको उत्पाद ऑब्जेक्ट के माध्यम से काम करना होगा। डेटा पढ़ें, आप उत्पाद वस्तु को एक संदेश भेजेंगे। उत्पाद वस्तु तब पढ़ी जाएगी मान और यह बताकर संदेश भेजें कि मूल्य क्या है। उत्पाद ऑब्जेक्ट परिभाषित करता है कि कौन से संचालन उत्पाद डेटा पर प्रदर्शन किया जा सकता है। यदि आप डेटा और उत्पाद ऑब्जेक्ट को संशोधित करने के लिए एक संदेश भेजते हैं निर्धारित करता है कि यह एक वैध अनुरोध है, यह आपके लिए ऑपरेशन करेगा और इसके साथ एक संदेश परिणाम वापस भेजेगा।

आप अपने दैनिक जीवन में हर समय इनकैप्सुलेशन का अनुभव करते हैं। एक मानव संसाधन विभाग के बारे में सोचो। वे कर्मचारियों के बारे में जानकारी छिपाते हैं। वे निर्धारित करते हैं कि यह डेटा कैसे है इस्तेमाल किया जा सकता है और हेरफेर किया जा सकता है। कर्मचारी डेटा के लिए कोई भी अनुरोध या डेटा को अपडेट करने का अनुरोध करना होगा उनके माध्यम से चला गया। एक अन्य उदाहरण नेटवर्क सुरक्षा है। सुरक्षा जानकारी या ए के लिए कोई अनुरोध नेटवर्क सुरक्षा व्यवस्थापक के माध्यम

से सुरक्षा नीति में परिवर्तन किया जाना चाहिए। सुरक्षा डेटा है नेटवर्क के उपयोगकर्ताओं से समझाया गया।

डेटा एनकैप्सुलेट करके आप अपने सिस्टम के डेटा को अधिक सुरक्षित और विश्वसनीय बनाते हैं। तुम्हे पता है कैसै डेटा एक्सेस किया जा रहा है और डेटा पर कौन से ऑपरेशन किए जा रहे हैं। यह program बनाता है रखरखाव बहुत आसान है और डिबगिंग प्रक्रिया को भी सरल करता है। आप संशोधित भी कर सकते हैं डेटा पर काम करने के लिए उपयोग की जाने वाली methods, और यदि आप परिवर्तन नहीं करते हैं कि method का अनुरोध कैसे किया जाता है और किस प्रकार का है प्रतिक्रिया वापस भेजी गई, फिर आपको method का उपयोग करके अन्य ऑब्जेक्ट को बदलने की आवश्यकता नहीं है। कब के बारे में सोचो आप मेल में एक पत्र भेजते हैं। आप पत्र देने के लिए डाकघर से अनुरोध करते हैं। पोस्ट कैसी?

कार्यालय इसे पूरा नहीं करता है। यदि यह पत्र को भेजने के लिए इसका उपयोग करने वाले मार्ग को बदलता है, तो यह नहीं करता है प्रभावित करें कि आप पत्र भेजने की पहल कैसे करते हैं। आपको डाकघर का आंतरिक पता करने की आवश्यकता नहीं है पत्र देने के लिए उपयोग की जाने वाली प्रक्रियाएं।

Polymorphism

Polymorphism अपने आप में एक ही अनुरोध संदेश का जवाब देने के लिए दो अलग–अलग अनोखा तरीका वस्तुओं की क्षमता है। उदाहरण के लिए, मैं अपने कुत्ते को कमांड की छाल और मेरे पक्षी को प्रतिक्रिया देने के लिए कमांड चिरप को प्रशिक्षित कर सकता था। दूसरी ओर, मैं उन्हें दोनों कमांड कमांड के जवाब के लिए प्रशिक्षित कर सकता था।

Polymorphism के माध्यम से मुझे पता है कि कुत्ता एक bark के साथ प्रतिक्रिया करेगा और पक्षी एक chirp के साथ जवाब देगा। यह ओओपी से कैसे संबंधित है? आप उन वस्तुओं को बना सकते हैं जो अपने आप

में एक ही अद्वितीय कार्यान्वयन संदेश का जवाब देते हैं। उदाहरण के लिए, आप किसी प्रिंटर ऑब्जेक्ट पर प्रिंट संदेश भेज सकते हैं प्रिंटर पर टेक्स्ट प्रिंट करें, और आप एक स्क्रीन ऑब्जेक्ट के लिए एक ही संदेश भेज सकते हैं जो प्रिंट करेगा आपके कंप्यूटर स्क्रीन पर एक विंडो के लिए पाठ।

Polymorphism का एक और अच्छा उदाहरण अंग्रेजी भाषा में शब्दों का उपयोग है। शब्द कई अलग–अलग अर्थ, लेकिन वाक्य के संदर्भ के माध्यम से आप यह घटा सकते हैं कि कौन सा अर्थ है इरादा है। तुम्हें पता है कि कोई कहता है कि "मुझे छुट्टी दो!" आपको उसका पैर तोड़ने के लिए नहीं कह रहा है!

ओओपी में आप इस प्रकार के Polymorphism को एक प्रक्रिया के माध्यम से लागू करते हैं जिसे ओवरलोडिंग कहा जाता है। आप ऐसा कर सकते हैं एक वस्तु के विभिन्न तरीकों को लागू करना जिनका एक ही नाम हो। वस्तु तब बता सकती है जो संदर्भ के आधार पर लागू करने की method (दूसरे शब्दों में, संख्या और प्रकार के तर्क संदेश का पारित)। उदाहरण के लिए, आप इन्वेंट्री ऑब्जेक्ट के दो तरीकों को देख सकते हैं।

एक उत्पाद की कीमत। इन दोनों methods को getPrice नाम दिया जाएगा। कोई अन्य वस्तु इस method को कॉल कर सकती है और या तो उत्पाद का नाम या उत्पाद आईडी पास करें। इन्वेंट्री ऑब्जेक्ट जो बता सकता है getPrice method से अनुरोध करने के लिए स्ट्रिंग मान या पूर्णांक मान पारित किया गया था।

Inheritance

अधिकांश वस्तुओं को श्रेणीबद्धता के अनुसार वर्गीकृत किया गया है। उदाहरण के लिए, आप सभी कुत्तों को एक साथ वर्गीकृत कर सकते हैं चार पैर और फर जैसे कुछ सामान्य लक्षण होने। उनकी नस्लें आगे वर्गीकृत करती हैं। उन्हें उप–श्रेणियों में आकार और डिमोरर जैसे सामान्य विशेषताओं के साथ। आप वस्तुओं का वर्गीकरण भी करते हैं उनके कार्य

के अनुसार। उदाहरण के लिए, वाणिज्यिक वाहन और मनोरंजक वाहन हैं। वहाँ ट्रक और यात्री कार हैं। आप कारों को उनके मेक और मॉडल के अनुसार वर्गीकृत करते हैं। की समझ बनाने के लिए दुनिया, आपको ऑब्जेक्ट पदानुक्रम और वर्गीकरण का उपयोग करने की आवश्यकता है।

आप अपने programs में वस्तुओं को सामान्य के अनुसार वर्गीकृत करने के लिए OOP में विशेषताएं और कार्य inheritance का उपयोग करते हैं। यह वस्तुओं के साथ काम करना आसान और अधिक सहज बनाता है। यह भी प्रोग्रामिंग को आसान बनाता है क्योंकि यह आपको एक माता–पिता में सामान्य विशेषताओं को संयोजित करने में सक्षम बनाता है।

ऑब्जेक्ट और इन ऑब्जेक्ट्स को चाइल्ड ऑब्जेक्ट्स में इनहेरिट करें। उदाहरण के लिए, आप एक कर्मचारी को परिभाषित कर सकते हैं ऑब्जेक्ट जो आपकी कंपनी में कर्मचारियों की सभी सामान्य विशेषताओं को परिभाषित करता है। फिर आप एक परिभाषित कर सकते हैं प्रबंधक ऑब्जेक्ट जो कर्मचारी ऑब्जेक्ट की विशेषताओं को inheritance में देता है लेकिन विशेषताओं को भी जोड़ता ळे आपकी कंपनी में प्रबंधकों के लिए अद्वितीय। प्रबंधक ऑब्जेक्ट स्वचालित रूप से किसी भी परिवर्तन को कर्मचारी वस्तु का कार्यान्वयन प्रतिबिंबित करेगा।

Aggregation

Aggregation तब होता है जब एक वस्तु में अन्य वस्तुओं का एक सममिश्रण होता है जो एक साथ काम करते हैं। उदाहरण के लिए, आपकी लॉन घास काटने की मशीन पहिया ऑब्जेक्ट्स, इंजन ऑब्जेक्ट, ब्लेड ऑब्जेक्ट, और इसी तरह का एक संयुक्त पर है। वास्तव में, इंजन ऑब्जेक्ट कई अन्य वस्तुओं का एक संयोजन है। के कई उदाहरण हैं हमारे आसपास की दुनिया में Aggregation। OOP में Aggregation का उपयोग करने की क्षमता एक शक्तिशाली विशेषता ळे आपको अपने programs में व्यवसाय प्रक्रियाओं को सही ढंग से लागू करने और कार्यानवित करने में सक्षम बनाता है।

THE HISTORY OF C#

1980 के दशक में, विंडोज ऑपरेटिंग सिस्टम पर चलने के लिए लिखे गए अधिकांश एपलिकेशन C में लिखे गए थे। भले ही C++ एक OOP भाषा है, लेकिन यकीनन यह मास्टर करने के लिए एक कठिन भाषा है और प्रोग्रामर है। Memory प्रबंधन और सुरक्षा जैसे हाउसकीपिंग कार्यों से निपटने के लिए जिम्मेदार। इन हाउसकीपिंग कार्यों को लागू करना मुश्किल है और अक्सर उपेक्षित किया जाता है जिसके परिणामस्वरूप छोटी गाड़ी के आवेदन मिलते हैं कि परीक्षण और बनाए रखने के लिए मुश्किल हैं।

1990 के दशक में, जावा प्रोग्रामिंग भाषा लोकप्रिय हो गई। क्योंकि यह एक प्रबंधित प्रोग्रामिंग भाषा है, यह प्रोग्रामर को हाउसकीपिंग कोड के बारे में चिंता करने से राहत देती है। प्रबंधित भाषाएं एक सामान्यीकृत तरीका प्रदान करती हैं (सामान्य कक्षाओं के आधार सेट के माध्यम से) Memory प्रबंधन और कचरा संग्रह जैसे हाउसकीपिंग विवरण। यह प्रोग्रामर को अनुमति देता है व्यापार तर्क पर ध्यान केंद्रित करने और त्रुटि–प्रवणता के बारे में चिंता करने से उन्हें मुक्त करता है हाउसकीपिंग कोड। परिणामस्वरूप, प्रोग्राम अधिक कॉम्पैक्ट, विश्वसनीय और डीबग करने में आसान होते हैं।

जावा की सफलता और इंटरनेट की बढ़ती लोकप्रियता को देखते हुए, Microsoft ने अपना विकास किया प्रबंधित प्रोग्रामिंग भाषाओं का सेट। Microsoft Windows वेब–आधारित दोनों applicationों को विकसित करना आसान बनाना चाहता था। ये प्रबंधित भाषाएं इस पर निर्भर करती हैं। .NET Framework सभी applicationों में आवश्यक हाउसकीपिंग कोड को निष्पादित करने के लिए बहुत अधिक कार्यक्षमता प्रदान करता है।

(.NET Framework)नेट फ्रेमवर्क, विकास के दौरान class libraries को C# नामक एक नई भाषा में लिखा गया था। C# का प्रमुख डिज़ाइनर

और लीड आर्किटेक्ट एंडर्स हेजेल्सबर्ग है।हेजेल्सबर्ग पहले इसके साथ शामिल थे टबी पास्कल और डेल्फी का डिज़ाइन। उन्होंने ओओपी भाषा डिजाइन करने के लिए इस पिछले अनुभव का लाभ उठाया कि इन भाषाओं की सफलताओं पर बनाया गया और उनकी कमियों में सुधार हुआ। हेजलबर्ग भी C और Java डेवलपर्स से अपील करने के लिए भाषा में C के समान सिंटैक्स शामिल किया गया है। बनाने के लक्ष्यों में से कुछ .नेट फ्रेमवर्क (.NET Framework), आम भाषा रनटाइम (सीएलआर), और सी# भाषा को आधुनिक अवधारणाओं जैसे कि object orientation, type safety, garbage collection का परिचय देना था, और मंच में सीधे संरचित Exceptions हैंडलिंग।

Microsoft का एक और लक्ष्य हमेशा प्रोग्रामर उत्पादकता में वृद्धि करना रहा है। अपने प्रारंभिक के बाद से 2002 में रिलीज़, Microsoft ने सुधार और नवाचार जारी रखा है।उनके साथ नेट फ्रेमवर्क कोर ढाँचा फ्रेम के शीर्ष पर निर्मित – C# और विजुअल बेसिक। Microsoft भी करने के लिए प्रतिबद्ध प्रदान कर रहा है। .NET डेवलपर्स एक अत्यधिक उत्पादक और सहज प्रोग्रामिंग के लिए आवश्यक उपकरण अनुभव। C# 4.0 की वर्तमान रिलीज़ के साथ और विजुअल स्टूडियो 2010, माइक्रोसॉफ्ट ने बहुत वृद्धि की है डेवेलपर्स के लिए भाषा और डिज़ाइन टाइम डेवलपिंग का अनुभव। जैसे–जैसे आप अपना काम करेंगे इस पुस्तक के माध्यम से, मुझे लगता है कि आप विजुअल स्टूडियो की शक्ति और उत्पादकता की सराहना करेंगे।

Summary

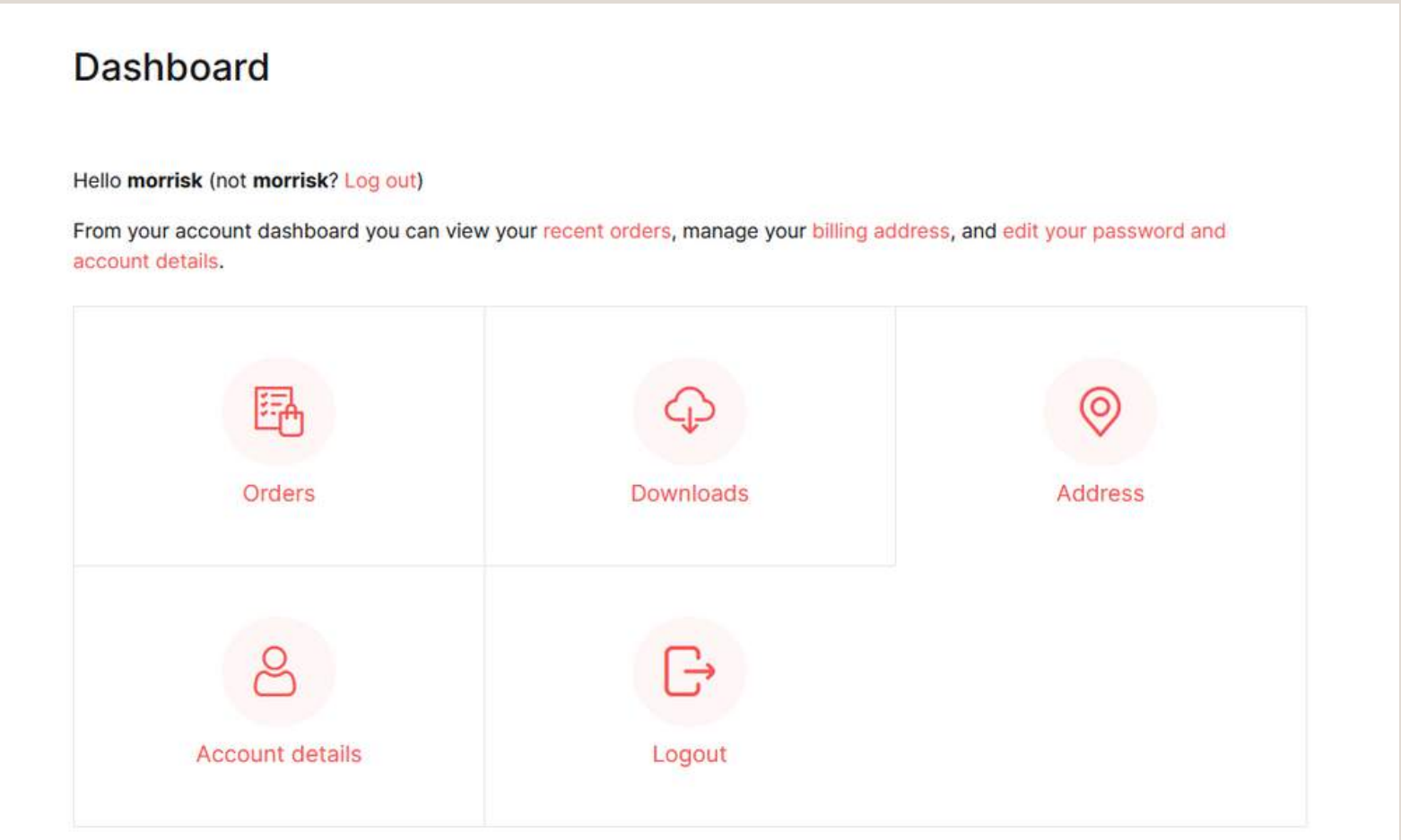
इस अध्याय में, आपको OOP से परिचित कराया गया और C# का संक्षिप्त इतिहास मिला। अब जब आप समझते हैं कि OOP भाषा का गठन क्या है और OOP भाषाएँ इतनी महत्वपूर्ण क्यों हैं? एंटरप्राइज़–स्तरीय application विकास, आपका अगला चरण ओओपी applications के डिज़ाइन के साथ परिचित होना है। उपयोगकर्ताओं की आवश्यकताओं

को पूरा करने के लिए, सफल applications को सावधानीपूर्वक योजनाबद्ध और विकसित किया जाना चाहिए।

अगला अध्याय ऑब्जेक्ट–ओरिएंटेड applications को डिज़ाइन करते समय उपयोग की जाने वाली कुछ तकनीकों से आपको परिचित कराने के उद्देश्य से तीन की श्रृंखला में पहला है। आप यह तय करने की प्रक्रिया देखेंगे कि किन वस्तुओं को एक application में शामिल किया जाना चाहिए और इन वस्तुओं की कौन सी विशेषताएँ उस application की कार्यक्षमता के लिए महत्वपूर्ण हैं।

Machhindra K
UPI ID: machhi29-2@okicici

उसके बाद आप **My Account** पेज पर जाकर **Download** में Ebooks डाउनलोड कर सकते है।



Thank you